Improved Competitive Ratios of Online Buffer Management Algorithms for Multi-Queue Switches in QoS Networks

AUTHOR(S):

Kobayashi, Koji M.; Miyazaki, Shuichi; Okabe, Yasuo

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
COMP2008-33 (2008-09)

QoSネットワーク上のマルチキュースイッチにおけるオンラインバッファ管理アルゴリズムの競合比の改良析

小林　浩二†　　宮崎　修一††　　岡部　寿男††

† 京都大学 情報学研究科　〒606–8501 京都府京都市左京区吉田本町
†† 京都大学 学術情報メディアセンター　〒606–8501 京都市左京区吉田本町
E-mail: †kobaya@net.ist.i.kyoto-u.ac.jp, ††{shuichi,okabe}@media.kyoto-u.ac.jp

あらまし　オンラインバッファ管理問題は, 近年のネットワークにおける主要な論点となっている QoS (Quality of Service) 保証実現のための, スイッチのキュー管理をオンライン問題として定式化した問題であり, 様々なモデルが考案されている. 本論文では Azar らによって考案された QoS ネットワーク上のマルチキュースイッチを扱ったモデルを取り上げる. Azar らは本モデルの競合比の上限を得るために, “the relaxed model” というモデルを導入している. 我々は relaxed model に対するオンラインアルゴリズム *DS* を提案することにより競合比を改良し, その結果としてマルチキュースイッチモデルの競合比の改良を行った. 以下は本論文におけるマルチキュースイッチモデルの競合比の改良の一例である. (i) プリエンプション不可能, かつパケットの価値が 2 値であるモデルにおいて, 十分大きな B に対して, 決定性アルゴリズムの競合比の上限を 4 から 3.177 に改良した. B は各キューが同時に保持できるパケットの最大数である. (ii) プリエンプション不可能, かつパケットの価値が 2 値であるモデルにおいて, 十分大きな B に対して, 乱択アルゴリズムの競合比の上限が高々 $\frac{17}{2} - \sqrt{30} \simeq 3.023$ であることを示した.

キーワード　競合比解析, マルチキュースイッチ, バッファ管理

Improved Competitive Ratios of Online Buffer Management Algorithms for Multi-Queue Switches in QoS Networks

Koji KOBAYASHI†, Shuichi MIYAZAKI††, and Yasuo OKABE††

† Graduate School of Informatics, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto, 606–8501 Japan
†† Academic Center for Computing and Media Studies, Kyoto University, Yoshida-Hommachi, Sakyo-ku Kyoto 606–8501, Japan
E-mail: †kobaya@net.ist.i.kyoto-u.ac.jp, ††{shuichi,okabe}@media.kyoto-u.ac.jp

Abstract　The online buffer management problem formulates the problem of queuing policies of network switches supporting QoS (Quality of Service) guarantee. For this problem, a lot of models have been considered. Among others, we focus on multi-queue switches in QoS Networks proposed by Azar et alAzar et al introduced the relaxed model in order to achieve a good upper bound on the competitive ratio for this model. In this paper, we improve the competitive ratios of several multi-queue models by improving an upper bound for the relaxed model. We propose an online algorithm *DS* (Dual Scheduling) for the relaxed model. This algorithm works for (either preemptive or non-preemptive) 2-value model, but it uses as subroutines online algorithms for the non-preemptive unit-value model, which has been extensively studied. The performance of *DS* depends on the performance of the algorithms used as subroutines. The followings are a couple of examples of the improvement on the competitive ratios of multi-queue models using our result: (i) We improved the competitive ratio of deterministic algorithms for the non-preemptive 2-value model from 4 to 3.177 for large enough B. a switch can store up to B packets simultaneously. (ii) We proved that the competitive ratio of randomized algorithms for the non-preemptive 2-value model is at most $\frac{17}{2} - \sqrt{30} \simeq 3.023$ for large enough B.

Key words　Competitive analysis; Multi-Queue Switches; Buffer management

1. Introduction

A great amount of work has been done in order to guarantee Quality of Service (QoS) on the Internet. One possible way of supporting QoS is differentiated services (DiffServ), where a traffic descriptor assigns a value to each packet according to the importance of the packet. QoS switches then try to decide acceptance/rejection and/or the order of transmission of packets using priority values. The goal of the buffer management algorithm is to maximize the total value of transmitted packets.

Recently, this kind of problem is modeled as online problems, and a great amount of work has been done. There have been proposed a lot of models, and the most basic one is the following [1]: A switch has a buffer of bounded size B. An input is a sequence of events. Each event is an arrival event or a send event. At an arrival event, one packet arrives at an input port. Each packet has the priority value and the size (the size is always one in this simplest case). A switch can store packets provided that the total size of stored packets does not exceed B, namely, a switch can store up to B packets simultaneously. At an arrival event, if the buffer is full, the new packet is rejected. If there is a room for the new packet, an online policy determines, without knowledge of the future, to accept it or not. At each send event, the packet at the head of the queue is transmitted. The goal of the problem is to maximize the sum of the values of transmitted packets. A goodness of an online policy is evaluated by the competitive analysis [9], [18]. If, for any input σ, an online policy A obtains value at least $1/c$ of the optimal offline policy for σ, then we say that A is c-competitive.

Up to the present, several models have been considered. Among them, Azar et al. have introduced a Multi-Queue Switches model [7]. In this model, a switch consists of m input ports and one output port, and each packet has a destination port. Each port has a buffer (FIFO queue), which can simultaneously store up to B packets. An input is a sequence of events. Each event is an arrival event or a scheduling event (which is similar to the send event described above). When a packet arrives at an arrival event, an online policy determines to accept it (if the buffer has room for the new packet), reject it, or preempt (namely, drop packets already in the buffer to make space) and accept the new packet. (We consider both models in which preepmtion is allowed and models not.) At a scheduling event, an online policy selects one nonempty buffer and transmits the first packet of the queue through the output port.

Previous Results. Several results on the competitiveness of the multi-queue model have been presented [2], [6]~[8], [11], [12], [17]. Table 1 summarizes current best upper and lower bound results for several models. In the multi-value model, $\alpha(\geq 1)$ is the ratio between the largest and the smallest values of packets. Among them, let us briefly review the technique in [7], which we improve in this paper.

In [7], the authors proposed a technique to convert an online algorithm for a single queue model into that of multi-queue model, so that the competitive ratio of the latter is at most twice that of the former. More formally, they defined the *relaxed model* of the multi-queue switch model (which will be formally defined in Sec. 2. 2). They showed that if (i) the competitive ratio of the single queue model is at most c, and (ii) the competitive ratio of the preemptive relaxed model is at most c', then the competitive ratio of the corresponding multi-queue model is at most cc'. They proved that the competitive ratio of a greedy algorithm for the relaxed model is at most 2, and combining this with the results for the single-queue models (Table 2), they obtained upper bounds described in Table 1.

Our Results. In this paper, we improve an upper bound on the competitive ratio of the preemptive relaxed model. We propose an algorithms $DS(A_1)$ for the preemptive relaxed model, where A_1 is an online algorithm for the unit-value multi-queue model that are used as subroutines. We prove that if the competitive ratio of A_1 is at most c, then the competitive ratio of $DS(A_1)$ is at most $c + \frac{2-c}{\alpha(2-c)+c-1}$. Using this result, we improve upper bounds on the competitive ratios of multi-queue 2-value models where the value of packets is restricted to 1 and $\alpha(\geq 1)$, as summarized in Table 1 (Details are included in Sec. 4.).

Note that Azar et al. [7] showed that improving competitive ratios for single-queue models implies improving competitive ratios for multi-queue models. Our results in this paper gives additional potential: Improving competitive ratios for unit-value multi-queue models also implies improving competitive ratios for 2-value multi-queue models.

Related Results. For the unit-value multi-queue model, a lot of works have been done. Azar et al. [7] gave a lower bound $1.366 - \Theta(1/m)$ of deterministic algorithms for any B, and an upper bound $\frac{e}{e-1}(\simeq 1.581)$ of a randomized algorithm. Albers et al. [2] showed that no greedy algorithm can be better than $2 - 1/B - \Theta(m^{-1/(2B-2)})$ for any B and large enough m. They also gave a $17/9(\simeq 1.89)$-competitive deterministic algorithm for $B \geq 2$, and it is optimal in the case $B = 2$. Furthermore, a lower bound $\frac{e}{e-1}(\simeq 1.581)$ of online deterministic algorithms for any B and large enough m, and a lower bound 1.465 of online randomized algorithms for any B and large enough m were presented. Azar et al. [6] showed a $\frac{e}{e-1}(\simeq 1.58)$-competitive deterministic al-

表 1 Competitve ratios for the multi-queue models

		Non-Preemptive		Preemptive	
		Lower Bound	Upper Bound	Lower Bound	Upper Bound
deterministic algorithm	2-value	$1+\frac{1}{\alpha \ln(\alpha/(\alpha-1))}$ [11]	$4-\frac{2}{\alpha}$ [7]	$\frac{e}{e-1} \approx 1.58$ [2]	2.564*, 2.6† [7]
			3.177‡ [this paper]		2.465‡ [this paper]
			3.778§ [this paper]		2.577§ [this paper]
	multi-value	$\ln(\alpha)+1$ [5]	$2\ln(\alpha)+4$ [7]	$\frac{e}{e-1} \approx 1.58$ [2]	$3-1/\alpha$ [12]
randomized algorithm	2-value	1.465 [2]		1.465 [2]	2.5 [7]
			3.023† [this paper]		2.214‡ [this paper]
					2.297† [this paper]
	multi-value	1.465 [2]		1.465 [2]	

*$B \to \infty$, †any B, ‡large enough B, §$B \geq 2$

表 2 single-queue model

		Non-Preemptive		Preemptive	
		Lower Bound	Upper Bound	Lower Bound	Upper Bound
deterministic algorithm	2-value	$2-1/\alpha$ [1]	$2-1/\alpha$ [5]	1.28 [13], [19]	1.282 [10]
	multi-value	$\ln(\alpha)+1$ [5]	$\ln(\alpha)+2$ [4]	1.419 [14]	1.732 [10]
randomized algorithm	2-value			1.197 [3]	$1+\alpha^{-\frac{1}{2}}-\alpha^{-1}$ [3]
	multi-value				

gorithm for $B > \log m$. Also, Schmidt [17] presented a 3/2-competitive randomized algorithm.

As for single-queue models, the current upper and lower bounds on competitive ratios are summarized in Table 2.

2. Preliminaries

In this section, we formally define the problem studied in this paper, and the relaxed model introduced in [7].

2.1 Online Buffer Management Problem for Multi-Queue Switches

A multi-queue switch has m input ports (FIFO queues) each of which is equipped with a buffer whose size is B. The size of a packet is one, and hence each port can store up to B packets simultaneously. Each packet has its *value* corresponding to the priority. In the unit-value model, the value of any packet is identical, say one. In the 2-value model, which is studied in this paper, each packet takes one of two values, say, 1 and $\alpha(\geq 1)$. We call a packet with value 1 (α, respectively) a 1-packet (an α-packet, respectively).

An input is a sequence of events. An *event* is an *arrival event* or a *scheduling event*. At an arrival event, a packet (say, p) arrives at an input port (1 through m), and the task of an online algorithm (or an online policy) is to select one of the following actions: insert an arriving packet into the corresponding queue (*accept* p), drop it (*reject* p), or drop a packet p' existing in the current buffer and accept p (*preempt* p'). (We consider in this paper both preemptive and non-preemptive models.) If a packet is accepted, it is stored at the tail of the corresponding input queue. We assume that no more than one packets arrive at the same time. At a scheduling event, an online algorithm selects one nonempty input port from m ones and transmits the packet at the head of the selected queue.

The *gain* of an algorithm is the sum of the values of transmitted packets, and our goal is to maximize it. The gain of an algorithm A for an input σ is denoted by $V_A(\sigma)$. If $V_A(\sigma) \geq V_{OPT}(\sigma)/c$ for an arbitrary input σ, we say that A is *c-competitive*, where OPT is an optimal offline policy for σ. Without loss of generality, we can assume that OPT never preempts packets. For simplicity of analysis, we consider the algorithm which transmits a packet at a scheduling event whenever its buffer is not empty. Such an algorithm is called *work-conserving*. (See [7], e.g.)

2.2 The Relaxed Model

The relaxed model is the same as the usual preemptive Multi-Queue model defined in Sec. 2.1, except for the following relaxation: In the original model, only a packet at the *head* of an input queue can be transmitted at a scheduling event, but in the relaxed model, any packet can be transmitted (namely, the buffer is not a queue). As is the case with Multi-Queue model, we can assume, without loss of generality, that OPT never preempts. Throughout this paper,

for simplicity, the 2-value multi-queue model (the unit-value multi-queue model and the preemptive relaxed model, respectively) is denoted by M_2 (M_1, and M_r, respectively). In addition, we denote OPT_2 (OPT_1 and OPT_r, respectively) optimal offline algorithms for M_2 (M_1 and M_r, respectively).

3. Algorithm DS

We propose Dual Scheduling Algorithm(DS) for M_r in Sec. 3. 1, and analyze the competitive ratio of DS in Sec. 3. 2.

3. 1 Dual Scheduling Algorithm(DS)

In this section, we give the definition of Dual Scheduling Algorithm (DS). Let A_1 be an online algorithm for M_1. DS uses A_1 as a subroutine, and hence it is written as $DS(A_1)$, but for simplicity, we write "DS" instead of "$DS(A_1)$" when A_1 is clear.

We give some definitions. For a time t when an event occurs, $t-$ represents a moment before t and after the previous event occurred. Similarly, $t+$ is a moment after t and before the next event occurs. The jth queue of the switch is denoted as $Q^{(j)}(1 \leqq j \leqq m)$. For an algorithm A_r for M_r, $h^{(j)}_{A_r}(t)$ denotes the number of packets A_r holds in $Q^{(j)}$ at time t when no event happens. $g^{(j)}_{DS}(t)$ denotes the number of α-packets DS holds in $Q^{(j)}$ at time t when an event does not happen. Let $\sigma(t)$ denote the prefix of σ up to time t. To define an algorithm, we need to specify its buffer management policy at an arrival event, and a scheduling policy at a scheduling event.

Buffer Management: DS accepts packets greedily, namely, when a 1-packet p arrives at $Q^{(i)}$ at time t, DS accepts p if $h^{(i)}_{DS}(t-) < B$. Otherwise, p is rejected. When an α-packet q arrives at $Q^{(i)}$ at t, if $h^{(i)}_{DS}(t-) < B$, then DS accepts q. If $h^{(i)}_{DS}(t-) = B$ and $g^{(i)}_{DS}(t-) < B$, a 1-packet is preempted and q is accepted. Otherwise, q is rejected.

Scheduling: $DS(A_1)$ uses two subroutines $AS(A_1)$ (standing for *α-packet Scheduling algorithm*) and $OS(A_1)$ (standing for *1-packet Scheduling algorithm*) defined later. (Hence A_1 is actually a *subsubroutine* of DS.) For simplicity, we write AS and OS instead of $AS(A_1)$ and $OS(A_1)$, respectively, when A_1 is clear.

At a scheduling event at time t, execute one of the following cases:

Case D1.1:

If there exists a queue $Q^{(i)}$ where $g^{(i)}_{DS}(t-) > 0$, DS calls AS, decides the α-packet p to be transmitted, and transmits p.

In this case, we say that "AS returns p" for convenience.

Execute one of the following cases.

Case D1.1.1:

If DS has no packet in its buffers after an executiton of Case D1.1, DS calls OS.

Note that OS does not return a packet by the definition of Step Ø3 on OS (See below). This case is executed in order to transmit all packets which OS stores but DS does not store after the execution of Case D1.1. Note that the sum of packets transmitted by OS is different from the sum of packets returned by OS.

Case D1.1.2:

If DS has a packet in its buffers after an executiton of Case D1.1.1, finish the execution.

Case D1.2:

If there does not exist a queue $Q^{(i)}$ where $g^{(i)}_{DS}(t-) > 0$, DS calls OS, decides the 1-packet p to be transmitted, and transmits p. Similarly, we say that "OS returns p".

Execute one of the following cases.

Case D1.2.1:

If DS has no packet in its buffers after an executiton of Case D1.2, DS calls OS.

This purpose of this case is similar to Case D1.1.1.

Case D1.2.2:

If DS has a packet in its buffers after an executiton of Case D1.2.1, finish the execution.

AS and OS are defined in the following:

α-Packet Scheduling Algorithm($AS(A_1)$):

(AS is called at time t.)

Step A1:

AS transforms $\sigma(t)$ into $\sigma'(t)$ by removing all arrival events of 1-packets from $\sigma(t)$.

Step A2:

AS simulates A_1 on $\sigma'(t)$, regarding $\sigma'(t)$ as an input for M_1. Let p be the packet that A_1 decides to transmit at the current scheduling event (namely, at the end of $\sigma'(t)$). Then, AS returns p to DS, and this routine is finished. Note that DS holds p at $t-$ since DS greedily accepts arriving α-packets, DS transmits α-packets by priority, and DS has transmitted the same packet as AS whenever AS was called.

1-Packet Scheduling Algorithm($OS(A_1)$):

(OS is called at time t.)

Step Ø1:

OS converts $\sigma(t)$ into $\sigma''(t)$ by removing all scheduling events where OS is not called by DS before t.

Note that an event where either Case D1.1.1 or D1.2.1

is executed is not removed although AS is called at this time. (This property is used in the proof of Lemma 3.9.)

Step Ø2:

OS simulates A_1 on $\sigma''(t)$, regarding $\sigma''(t)$ as an input for M_1. However, since two kinds of packets, 1-packets and α-packets can arrive at an arrival event in $\sigma''(t)$, A_1 cannot be run for $\sigma''(t)$ if nothing is done. So A_1 executes a buffer management forarriving packets according to the following definition. We will give one more remark to define OS. $h^{(i)}_{OS}(t)$ also denotes the number of packets which OS holds in $Q^{(i)}$ at time t when no event happens.

Buffer Management for OS: OS accepts 1-packets greedily, namely, when an α-packet q arrives at $Q^{(i)}$ at time t'', OS accepts q if $h^{(i)}_{OS}(t''-) < B$. Otherwise, q is rejected. When a 1-packet p arrives at $Q^{(i)}$ at t'', if $h^{(i)}_{OS}(t''-) < B$, then OS accepts p. If $h^{(i)}_{OS}(t''-) = B$ and there exists an α-packet q' in $Q^{(i)}$, q' is preempted and p is accepted. If $h^{(i)}_{OS}(t''-) = B$, there does not exist an α-packet q' in $Q^{(i)}$ and DS accepts, OS preempts q'' which DS does not hold at $Q^{(i)}$ at $t-$, and accepts p Otherwise, p is rejected.

Let p be the packet that A_1 decides to transmit at the current scheduling event (namely, at the end of $\sigma'(t)$).

Step Ø3:

If the buffer of OS is empty at the end of $\sigma''(t)$, OS does not return any packet and this routine is finished. Otherwise, let $Q^{(i)}$ be the queue where A_1 selects a packet to transmit at the current scheduling event (namely, at the end of $\sigma''(t)$). OS selects an arbitrary packet p from $Q^{(i)}$, and performs one of the following cases depending on p.

Case Ø3.1:

If DS holds p in the buffer at $t-$, OS transmits p and returns p to DS. This routine is finished.

Case Ø3.2:

If DS does not hold p in the buffer at $t-$, OS transmits p. Go back to Step Ø3.

Note that OS returns at most one packet but can transmit some packets at a single scheduling event. Also, note that OS can return all 1-packets which are accepted and are not dropped by DS (See Lemma 3.8).

3.2 Competitive Analysis of DS

3.2.1 Overview of the Analysis

For an input σ_r for M_r, let $\mathcal{T}_{B,1}(\sigma_r)$ ($\mathcal{T}_{B,\alpha}(\sigma_r)$, respectively) be the number of 1-packets (α-packets, respectively) p such that (i) p arrives at $Q^{(i)}$ at time t where $g^{(i)}_{DS}(t-) = B$, (ii) DS drops p (namely, p is rejected at t since DS greedily accepts arriving packets), and (ii) OPT_r accepts p, which is eventually transmitted since OPT_r never preempts. For an input σ_r for M_r, let $\mathcal{T}_{\bar{B},1}(\sigma_r)$ ($\mathcal{T}_{\bar{B},\alpha}(\sigma_r)$, respectively) be the number of 1-packets (α-packets, respectively) p such that (i) p arrives at $Q^{(i)}$ at time t where $g^{(i)}_{DS}(t-) < B$, (ii) DS drops p, and (iii) OPT_r accepts p. Since DS accepts arriving packets greedily, if an α-packet is dropped from $Q^{(i)}$ at t, then $g^{(i)}_{DS}(t-) = B$. Therefore, $\mathcal{T}_{\bar{B},\alpha}(\sigma_r) = 0$ holds. We will prove in Lemma 3.2, that for any online algorithm A_r for M_r and for any input σ'_r for which the above defined $\mathcal{T}_{B,1}(\sigma'_r) > 0$, there exists another input σ''_r for which $\mathcal{T}_{B,1}(\sigma''_r) = 0$ and the competitive ratio of A_r is equal to or larger than that for σ'_r. Therefore, it suffices to consider only inputs σ_r for which $\mathcal{T}_{B,1}(\sigma_r) = 0$. Hence the numbers of 1-packets and α-packets, respectively, OPT_r accepts but DS drops are $\mathcal{T}_{\bar{B},1}(\sigma_r)$ and $\mathcal{T}_{B,\alpha}(\sigma_r)$. Then, $V_{OPT_r}(\sigma_r) \leqq V_{DS}(\sigma_r) + \mathcal{T}_{\bar{B},1}(\sigma_r) + \alpha\mathcal{T}_{B,\alpha}(\sigma_r)$ holds. Let $R_A(\sigma_r)$ ($A = \{AS, OS\}$) be the number of packets returned by A for an input σ_r for M_r. Note that DS transmits a packet returned by AS or OS. Then $V_{DS}(\sigma_r) = R_{OS}(\sigma_r) + \alpha R_{AS}(\sigma_r)$ by definition. Let D-event be a scheduling event where DS transmits an α-packet and OPT_r transmits a 1-packet. Let $\mathcal{K}$ be the number of D-events. Suppose that the competitive ratio of A_1, a subroutine of DS, is at most c (in M_1). In Sec. 3.2.2, we show that $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\} \geqq \mathcal{T}_{B,\alpha}(\sigma_r)$, and in Sec. 3.2.3, we prove that $R_{AS}(\sigma_r) + \min\{(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geqq \mathcal{T}_{B,\alpha}(\sigma_r) + \mathcal{T}_{\bar{B},1}(\sigma_r)$. Therefore, $V_{OPT_r}(\sigma_r) = R_{OS}(\sigma_r) + \alpha R_{AS}(\sigma_r) + \mathcal{T}_{\bar{B},1}(\sigma_r) + \alpha\mathcal{T}_{B,\alpha}(\sigma_r) \leqq \frac{\alpha c(2-c)+c^2-2c+2}{\alpha(2-c)+c-1} V_{DS}(\sigma_r)$. (See [15] how to calculate this ratio.) Hence, we have the following theorem:

[Theorem 3.1] If the competitive ratio of A_1 for M_1 is at most c, then the competitive ratio of $DS(A_1)$ is at most $\frac{\alpha c(2-c)+c^2-2c+2}{\alpha(2-c)+c-1}$.

3.2.2 Analysis of AS

At first, we show that it is sufficient to consider only inputs σ_r such that $\mathcal{T}_{B,1}(\sigma_r) = 0$. The proof of the following lemma is shown in [15].

[Lemma 3.2] Let σ_r for M_r be an input for which $\mathcal{T}_{B,1}(\sigma_r) > 0$. Then, there exists an input σ'_r for M_r for $\mathcal{T}_{B,1}(\sigma'_r) = 0$ such that $\frac{V_{OPT_r}(\sigma'_r)}{V_{DS}(\sigma'_r)} \geqq \frac{V_{OPT_r}(\sigma_r)}{V_{DS}(\sigma_r)}$.

For analysis, we give some definitions. Let A_1 be an online algorithm for M_1 and σ_1 be an input for M_1. Let t be a time when no event occurs. We call $h^{(i)}_{A_1}(t) - h^{(i)}_{OPT_1}(t)$ *gaps* at $Q^{(i)}$ at t for A_1, OPT_1 and σ_1 at M_1 if $h^{(i)}_{A_1}(t) - h^{(i)}_{OPT_1}(t) > 0$. For better understanding of gaps, we estimate the degree of increase and decrease of gaps in [15]. Then, we call an arrival event where A_1 drops a packet from $Q^{(i)}$ a *p-event* for A_1, OPT_1 and σ_1 at $Q^{(i)}$ at t. Also, for a p-event for A_1, OPT_1 and an input σ_1 at $Q^{(i)}$ at time t, the corresponding *g-event*

(gap-event) for A_1, OPT_1 and an input σ_1 is a scheduling event that happens at $Q^{(i)}$ at t' satisfying the following three inequalities: $h^{(i)}_{A_1}(t'') - h^{(i)}_{OPT_1}(t'') \geq h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-) = B - h^{(i)}_{OPT_1}(t-) \quad (\forall t'' \in [t'+, t-])$, $h^{(i)}_{A_1}(t'-) = h^{(i)}_{A_1}(t'+)$, and $h^{(i)}_{OPT_1}(t'-) = h^{(i)}_{OPT_1}(t'+) + 1$. An online algorithm A_1, an optimal offline algorithm OPT_1 and an input σ_1 for σ_r decide whether an event is a p-event (g-event, respectively) or not. Hence, if an event e is a p-event(g-event, respectively), we write e is a p-event for A_1, OPT_1 and σ_1 (g-event for A_1, OPT_1 and σ_1, respectively). We may omit A_1, OPT_1 or σ_1 when they are clear.

Here, we give some definitions about the number of p-events and g-events. For an input σ_1 at M_1, and an online algorithm A_1 for M_1, let $\mathcal{P}_{A_1}(\sigma_1)$ ($\mathcal{G}_{A_1}(\sigma_1)$, respectively) denote the number of p-events for A_1 and σ_1 (g-events for A_1 and σ_1, respectively). Note that AS and OS can be regarded as A_1 since they convert an input σ_r for M_r into σ_1 for M_1, and decide a packet to be transmitted by DS. The proof of the following lemma is shown in [15].

[Lemma 3.3] Let A_1 be an online algorithm for M_1 and σ_1 be an input for M_1. Then, $\mathcal{P}_{A_1}(\sigma_1) = \mathcal{G}_{A_1}(\sigma_1)$.

Here, we give some definitions. For an input σ_r for M_r and a time t when no event happens, let $\mathcal{T}_{B,\alpha}(\sigma_r, t)$ be the number of α-packets p such that (i) p arrives at $Q^{(i)}$ at time t' where $g^{(i)}_{DS}(t'-) = B$, (ii) DS drops p at $t''(\in [t', t])$, and (iii) OPT_r accepts p at t'. For an input σ_1 for M_1, and an online algorithm A_1 for M_1, let $\mathcal{P}_{A_1}(\sigma_1, t)$ denote the number of p-events for A_1 and σ_1 where happen before t. Note that A_1 can be AS or OS, which can be regarded as an online algorithm for M_1. For any model M_1, or M_r, let σ be an input and A be an algorithm. (Note that A includes OS, which is an algorithm in M_r.) Then define $T_A(\sigma)$ to be the number of transmitted packets by A for an input σ.

In Sec. 3. 2. 2, we show a relation between the number of α-packets which are not accepted by DS, namely $\mathcal{T}_{B,\alpha}(\sigma_r)$, and the number of p-events for AS, OPT_r and an input σ_r at M_r, namely, $\mathcal{P}_{AS}(\sigma_r)$. In addition, we show an upper bound of the number on g-events for AS, OPT_r and an input σ_r at M_r, namely, $\mathcal{G}_{AS}(\sigma_r)$. Now, we specify a gap for AS, OPT_r and an input σ_r at M_r. Let σ_r be an input for M_r. We call $g^{(i)}_{DS}(t) - h^{(i)}_{OPT_r}(t)$ gaps at $Q^{(i)}$ at t for AS, OPT_r and σ_r if $g^{(i)}_{DS}(t) - h^{(i)}_{OPT_r}(t) > 0$. Then, we call an arrival event where DS drops an α-packet from $Q^{(i)}$ a *p-event* for AS, OPT_r and σ_{AS} at $Q^{(i)}$ at t. Also, for a p-event at $Q^{(i)}$ at time t, the corresponding *g-event* (gap-event) for AS, OPT_r and σ_r is a scheduling event that happens at t' satisfying the following three conditions: $g^{(i)}_{DS}(t'') - h^{(i)}_{OPT_r}(t'') \geq g^{(i)}_{DS}(t-) - h^{(i)}_{OPT_r}(t-) = B - h^{(i)}_{OPT_r}(t-) \quad (\forall t'' \in [t'+, t-])$, $g^{(i)}_{DS}(t'-) = g^{(i)}_{DS}(t'+)$, and $h^{(i)}_{OPT_r}(t'-) = h^{(i)}_{OPT_r}(t'+) + 1$. The proof of the following lemma is shown in [15].

[Lemma 3.4] Let t be a time when no event happens, and σ_r be an input for M_r. Then, $\mathcal{P}_{AS}(\sigma_r, t) \geq \mathcal{T}_{B,\alpha}(\sigma_r, t)$.

Recall that a D-event is a scheduling event at which OPT_r transmits a 1-packet and AS transmits an α-packet. In order to evaluate the number of g-events when $\mathcal{K}$ D-events happen, we consider a modification of M_1, which we call the *sleep model* (denoted by M_s). An input for M_s is a sequence of events. An event is an arrival event, an N-scheduling event (normal scheduling event) or an S_{OPT}-scheduling event (OPT scheduling sleep event). An arrival event for M_s is the same as M_1, and an N-scheduling event is the same as a scheduling event for M_1. An S_{OPT}-scheduling event is an event in which an online algorithm A can transmit a packet from a queue, but OPT cannot. Furthermore, A for M_s cannot distinguish between an N-scheduling event and an S_{OPT}-scheduling event. For simplicity, we denote OPT_s an optimal offline algorithms for M_s. Then, we say that OPT_s *sleeps* for A_s if A_s transmits a packet at an S_{OPT}-scheduling event. Note that an online algorithm A_1 for M_1 can be used for M_s. We define p-events and g-events for A_1, OPT_s and an inpit σ_s at M_s in the same way as M_1. The proofs of the following lemmas are shown in [15].

[Lemma 3.5] Let A_1 be an online algorithm for M_1 whose competitive ratio is at most c. Let σ_s be any input for M_s in which OPT_s sleeps for A_1 exactly k times, and let σ_1 be an input for M_1 obtained from σ_s by replacing all S_{OPT}-scheduling events by N-scheduling events. Then, $\min\{(c-1)T_{A_1}(\sigma_1), T_{A_1}(\sigma_1) - k\} \geq \mathcal{G}_{A_1}(\sigma_s)$.

[Lemma 3.6] Let σ_r be an input for M_r. Then, $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\} \geq \mathcal{G}_{AS}(\sigma_r)$.

Proof. DS calls AS and transmits an α-packet, but OPT_r transmits a 1-packet at a D-event. So, we can consider OPT_r sleeps for DS, namely, AS at a D-event. Therefore, using Lemma 3.5, the number of g-events for AS, OPT_r and σ_r is at most $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\}$. □

Now, we are ready to show the main lemma in this section. The proof of the following lemma is shown in [15].

[Lemma 3.7] Let σ_r be an input for M_r. Then, $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\} \geq \mathcal{T}_{B,\alpha}(\sigma_r)$.

3. 2. 3 Analysis of OS

In this section, we analyze OS to evaluate the number of packets which OPT_r transmits but OS cannot return (Note that the sum of packets returned by OS is different from the sum of packets transmitted by OS). At first, we show lemmas about properties of packets which OS and DS store at the same time. The proof of the following lemmas is shown in [15].

[Lemma 3.8] Let t be a time when no event occurs. If DS stores a 1-packet p at $Q^{(i)}$ at t, OS also stores p at $Q^{(i)}$ at t.

[Lemma 3.9] Let t be a time when no event happens. Then, $\forall i\ h^{(i)}_{OS}(t) \geqq h^{(i)}_{DS}(t)$.

We give some definitions. For an input σ for M_r, $A = \{AS, OS\}$, and a time t when an event does not happen, $R_A(\sigma, t)$ denotes the number of packets returned by A before t, and $T_{OS}(\sigma, t)$ denotes the number of packets transmitted by OS before t. The proofs of the following lemma and corollary are shown in [15].

[Lemma 3.10] Let t be a time when no event occurs, and σ_r be an input for M_r. Then, $\forall t\ R_{AS}(\sigma_r, t) + R_{OS}(\sigma_r, t) + \sum_{i=1}^{m} h^{(i)}_{DS}(t) \geqq T_{OS}(\sigma_r, t) + \sum_{i=1}^{m} h^{(i)}_{OS}(t)$.

[Corollary 3.11] Let σ_r be an input for M_r. Then, $R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \geqq T_{OS}(\sigma_r)$.

We give a definition for the following lemma. Let t be a time when no event happens, and σ_r be an input for M_r. $\mathcal{T}_{\bar{B},1}(\sigma_r, t)$ denotes the number of 1-packets p such that (i) p arrives at $Q^{(i)}$ at time t' where $g^{(i)}_{DS}(t'-) < B$ and DS drops p at $t''(\in [t', t])$, and (ii) OPT_r accepts p at t'. Also, we define a p-event and a g-event for an online algorithm OS, OPT_r and an input σ_r for M_r in the same way as M_1. The proof of the following lemma is shown in [15].

[Lemma 3.12] Let t be a time when no event happens, and σ_r be an input for M_r. Then, $\mathcal{P}_{OS}(\sigma_r, t) \geqq \mathcal{T}_{B,\alpha}(\sigma_r, t) + \mathcal{T}_{\bar{B},1}(\sigma_r, t)$.

In order to evaluate an upper bound on the number of g-events for OS, OPT_r and an input σ_r at M_r, we consider an extension of M_s (say, $M_{s'}$). For simplicity, we denote $OPT_{s'}$ an optimal offline algorithm for $M_{s'}$. An input for $M_{s'}$ is a sequence of events. An event is an arrival event, an N-scheduling event (normal scheduling event), an S_{OPT}-scheduling event (OPT sleep scheduling event) or an S_{ON}-scheduling event (online algorithm sleep scheduling event). An arrival event, an N-scheduling event, and an S_{OPT}-scheduling event are the same to those for M_s, respectively. An S_{ON}-scheduling event is a counterpart to an S_{OPT}-scheduling event. Namely, an S_{ON}-scheduling event is an event where $OPT_{s'}$ can transmit a packet, but an online algorithm $A_{s'}$ for $M_{s'}$ cannot. Further, $A_{s'}$ cannot know the presence of any S_{ON}-scheduling events. Hence, an online algorithm A_1 for M_1 can be applied for $M_{s'}$ without modification. Then, we say that an online algorithm A_1 for $M_{s'}$ *sleeps for* $OPT_{s'}$ if A_1 holds a packet at $t-$, a scheduling event happens at t, and $OPT_{s'}$ transmits a packet at an S_{ON}-scheduling event. We define p-events and g-events for an online algorithm A_1, $OPT_{s'}$ and an input $\sigma_{s'}$ at $M_{s'}$ in the same way as M_1. Now, we are ready to show an upper bound on the number of g-events for an online algorithm A_1 for $M_{s'}$, $OPT_{s'}$ and an input $\sigma_{s'}$ for $M_{s'}$ at $M_{s'}$ in the following lemma. The proofs of the following lemma and corollary are shown in [15].

[Lemma 3.13] Let A_1 be an online algorithm for M_1 whose competitive ratio is at most c. Let $\sigma_{s'}$ be any input for $M_{s'}$ in which $OPT_{s'}$ sleeps for A_1 exactly k times, and A_1 sleeps for $OPT_{s'}$ exactly k' times, and let σ_1 be an input for M_1 obtained from $\sigma_{s'}$ by replacing all S_{OPT}-scheduling events by N-scheduling events (Note that we do not change S_{ON}-scheduling events). Then, $k' + \min\{(c-1)T_{A_1}(\sigma_1), T_{A_1}(\sigma_1) - k\} \geqq \mathcal{G}_{A_1}(\sigma_{s'})$.

Now, we are ready to show the lemma which evaluates the number of g-events for OS and OPT_r at M_r.

[Lemma 3.14] Let σ_r be an input for M_r. Then, $R_{AS}(\sigma_r) + \min\{(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geqq \mathcal{G}_{OS}(\sigma_r)$.

Proof. At first, we consider the number of g-events for OS, and σ_r which happen at a time when DS calls AS. At this event, OS cannot transmit a packet since by the way of modification of σ_r in Step Ø1 but OPT_r transmits a packet. Therefore, we can regard OS as sleeping for OPT_r at $R_{AS}(\sigma_r)$ scheduling events.

Secondly, we consider a scheduling event where DS calls OS in σ_r. OS can transmit $T_{OS}(\sigma_r)$ packets at $R_{OS}(\sigma_r)$ scheduling events but OPT_r transmits at most $R_{OS}(\sigma_r)$ packets at these events. Hence, we can regard OPT_r as $T_{OS}(\sigma_r) - R_{OS}(\sigma_r)$ times sleeping for OS. By these facts, and Lemma 3.13, $R_{AS}(\sigma_r) + \min\{(c-1)T_{OS}(\sigma_r), T_{OS}(\sigma_r) - (T_{OS}(\sigma_r) - R_{OS}(\sigma_r))\} \geqq \mathcal{G}_{OS}(\sigma_r)$. Since $R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \geqq T_{OS}(\sigma_r)$ using Corollary 3.11, $R_{AS}(\sigma_r) + \min\{(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geqq \mathcal{G}_{OS}(\sigma_r)$ holds. □

The proof of the following lemma is shown in [15].

[Lemma 3.15] Let σ_r an input for M_r. Then, $R_{AS}(\sigma_r) + \min\{(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geqq \mathcal{T}_{B,\alpha}(\sigma_r) + \mathcal{T}_{\bar{B},1}(\sigma_r)$.

4. Competitive Ratios for the Multi-Queue Model

In this section, we give upper bounds on several variants of M_2, using Theorem 3.1 and Theorem A.1 in [15], whose proofs are shown in [15].

[Corollary 4.1] There is an online deterministic algorithm for the non-preemptive M_2 whose competitive ratio is at most 3.177 for large enough B.

[Corollary 4.2] There is an online deterministic algorithm for the non-preemptive M_2 whose competitive ratio is at most 3.778 for $B \geqq 2$.

[Corollary 4.3] There is an online deterministic algorithm for the preemptive M_2 whose competitive ratio is at most 2.465 for large enough B.

[Corollary 4.4] There is an online deterministic algorithm for the preemptive M_2 whose competitive ratio is at most

2.577 for $B \geqq 2$.

[Corollary 4.5] There is an online randomized algorithm for the preemptive M_2 whose competitive ratio is at most 2.214 for large enough B.

[Corollary 4.6] There is an online randomized algorithm for the preemptive M_2 whose competitive ratio is at most 2.297 for any B.

[Corollary 4.7] There is an online randomized algorithm for the non-preemptive M_2 whose competitive ratio is at most 3.174 for any B.

5. Concluding Remarks

Although DS can use any algorithm as a subroutine, we conjecture that it can achieve a better competitive ratio if it is customized to one specific online algorithm.

References

[1] W. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosén, "Competitive queue policies for differentiated services," *Journal of Algorithms*, Vol. 55, No. 2, pp. 113–141, 2005,

[2] S. Albers and M. Schmidt, "On the performance of greedy algorithms in packet buffering," *SIAM J. Comput.*, Vol. 35, No. 2, pp. 278–304, 2005.

[3] N. Andelman, "Randomized queue management for DiffServ," *In Proc. of 17th annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 1–10, 2005.

[4] N. Andelman and Y. Mansour, "Competitive management of non-preemptive queues with multiple values," *In Proc. of 17th International Symposium on Distributed Computing*, pp. 166–180, 2003.

[5] N. Andelman, Y. Mansour and A. Zhu, "Competitive queueing policies for QoS switches," *In Proc. of 14th annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 761–770, 2003.

[6] Y. Azar and A. Litichevskey, "Maximizing throughput in multi-queue switches," Algorithmica, Vol.45, No. 1, pp, 69–90, 2006,

[7] Y. Azar and Y. Richter, "Management of multi-queue switches in QoS networks," Algorithmica, Vol.43, No. 1-2, pp, 81–96, 2005,

[8] Y. Azar and Y. Richter, "The zero-one principle for switching networks," *In Proc. of 36th annual ACM Symposium on Theory of Computing*, pp. 64–71, 2004.

[9] A. Borodin and R. El-Yaniv, "Online computation and competitive analysis," *Cambridge University Press*, 1998.

[10] M. Englert and M. Westermann, "Lower and upper bounds on FIFO buffer management in QoS switches," *In Proc. of 14th annual European Symposium on Algorithms*, pp. 352–363, 2006.

[11] T. Itoh and T. Nagumo, "Improved lower bounds for competitive ratio of multi-queue switches in QoS networks," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E88-A, No. 5, pp. 1155–1165, 2005.

[12] T. Itoh and N. Takahashi, "Competitive analysis of multi-queue preemptive QoS algorithms for general priorities," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E89-A, No. 5, pp. 1186–1197, 2006.

[13] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko, "Buffer overflow management in QoS switches," *SIAM J. Comput.*, Vol. 33, No. 3, pp. 563–583, 2004.

[14] A. Kesselman, Y. Mansour and R. Stee, "Improved competitive guarantees for QoS buffering," *In Proc. of 11th annual European Symposium on Algorithms*, pp. 361–372, 2003.

[15] K. Kobayashi, S. Miyazaki and Y. Okabe, "Improving Online Buffer Management for Multi-Queue Switches in QoS Networks," unpublished manuscript (http://www.net.ist.i.kyoto-u.ac.jp/kobaya/comp0809.pdf), 2008.

[16] Z. Lotker and B. Patt-Shamir, "Nearly optimal FIFO buffer management for two packet classes," *Computer Networks*, Vol. 42, No. 4, pp. 481–492, 2003,

[17] M. Schmidt, "Packet buffering: Randomization beats deterministic algorithms," *In Proc. of 22nd International Symposium on Theoretical Aspects of Computer Science*, pp. 293–304, 2005.

[18] D. Sleator and R. Tarjan, "Amortized efficiency of list update and paging rules," *CACM 28*, pp. 202–208, 1985.

[19] M. Sviridenko, "A lower bound for on-line algorithms in the FIFO model," unpublished manuscript, 2001.